2017년 데이터 통신

-02-

제출일자	2017.03.28.
이 름	정 윤 수
학 번	201302482
분 반	01

1. 실습 개요

(1)실습 일시 및 장소

: 이번 과제 실습은 3월 24일 기숙사 내방에서 수행이 되었다.

(2)실습 목적

: 이번 실습 과제의 목적은 IPC를 구현을 하여 두 개의 프로세스 간에 간단한 통신을 구현을 하는 것을 목적으로 하고 있다. 하나의 프로세스에서 입력을 실행하면 다른 하나의 프로세스에서 출력을 하는 것을 구현을 한다.

2. 프로토콜 스택

(1) BaseLayer.cpp

: 각 계층들의 기본이 되는 Layer로 다른 Layer들은 이 계층을 상속을 받아서 사용을 한다.

(2) IPCAPPDlg.cpp

: 채팅창이 구현이 되어있는 Layer로 맨처음에 다른 Layer들의 계층 구조를 형성을 시켜준 후 사용자가 입력을 실행을 하면 다음 계층인 ChapAppLayer로 받은 데이터를 전송을 해주거나 전송 받은 데이터를 화면에 출력을 해준다..

(3) ChapAppLayer.cpp

: EthenetLayer로 전송을 하거나 데이터를 받는 역할을 하는 계층으로 데이터를 전송을 할 때에는 데이터를 다시 가공해서 EthernetLayer로 전송을 하고 데이터를 수신을 할 때 에는 EthenetLayer에서 온 데이터가 자신의 것이 맞는지 확인을 가공을 한 후하고 IPCAPPDlg Layer로 보내준다.

(4) EthernetLayer.cpp

: ChapAppLayer에서 온 데이터를 FileLayer로 전송을 하거나 FileLayer에서 온 데이터를 ChapAppLayer로 전송을 하는 역할을 당담을 한다.

(5) FileLayer.cpp

: EthernetLayer에서 전송이 된 데이터를 공유 파일에 저장을 하거나 공유 파일에 저장이 되어진 데이터를 EthernetLayer에 보내는 역할을 수행을 한다.

3. 구현 설명

(1) BaseLayer.cpp

```
void CBaseLayer::SetUnderUpperLayer(CBaseLayer *pUULayer)
{
    if ( !pUULayer ) // if the pointer is null,
    {
#ifdef _DEBUG
        TRACE( "[CBaseLayer::SetUnderUpperLayer] The variable , 'pUULayer' is NULL" ) ;
#endif
        return ;
    }

    this->mp_UnderLayer = pUULayer;
    pUULayer->SetUpperLayer(this);
}
```

-매개변수로 받아온 계층을 현재 자신의 계층 아래 계층으로 설정을 하고 받아온 계층의 상위 계층을 지금 현재 이 함수를 호출을 한 계층으로 설정을 하는 함수이다. 매개변수로 들어온 Layer가 null이면 오류 메시지를 출력을 해주고 이 함수는 맨 처음 계층 구조를 형성을 할 때 사용이 되어진다.

```
void CBaseLayer::SetUpperUnderLayer(CBaseLayer *pUULayer)
{
    if ( !pUULayer ) // if the pointer is null,
    {
#ifdef _DEBUG
        TRACE( "[CBaseLayer::SetUpperUnderLayer] The variable , 'pUULayer' is NULL" ) ;
#endif
        return ;
    }
    SetUpperLayer(pUULayer);
    pUULayer->SetUnderLayer(this);
}
```

- 위에있는 함수와는 반대의 일은 하는 함수이다. 매개변수로 들어온 Layer를 자신의 상위 계층으로 만들고 그 계층의 하위 계층을 지금 이 함수를 호출을 한 Layer가 되게 만든다.

(2) IPCAppDlg.cpp

```
CIPCAppDlg::CIPCAppDlg(CWnd* pParent /*=NULL*/)
    : CDialog(CIPCAppDlg::IDD, pParent),
      CBaseLayer( "ChatDlg" ),
      m_bSendReady( FALSE ),
      m_nAckReady( -1 )
{
    //{{AFX_DATA_INIT(CIPCAppDlg)
    m_unDstAddr = 0;
    m_unSrcAddr = 0;
    m_stMessage = _T("");
    //}}AFX_DATA_INIT
    // Note that LoadIcon does not require a subsequent DestroyIcon in Win32
    m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);

    m_LayerMgr.AddLayer( new CChatAppLayer( "ChatApp" ) ) ;
    m_LayerMgr.AddLayer( new CEthernetLayer( "Ethernet" ) ) ;
    m_LayerMgr.AddLayer( new CFileLayer( "File" ) ) ;
    m_LayerMgr.AddLayer( this ) ;

    m_LayerMgr.ConnectLayers("File ( *Ethernet ( *ChatApp ( *ChatDlg ) ) ) ");

    m_ChatApp = (CChatAppLayer*) mp_UnderLayer ;
}
```

- 계층을 만드는 역할을 하는 함수이다. AddLayer함수를 이용을하여 LayerManager의 연결 리스트에 각각의 Layer들을 추가를 해준다 그 후 connectLayer() 함수를 이용을 하여 File Layer가 가장 하위에 있고 IPCApptDlg가 가장 상위에 있는 계층 구조를 형성을 한다.

```
BEGIN_MESSAGE_MAP(CIPCAppDlg, CDialog)
    //{{AFX_MSG_MAP(CIPCAppDlg)
    ON_WM_SYSCOMMAND()
    ON_WM_PAINT()
    ON_WM_QUERYDRAGICON()
    ON_BN_CLICKED(IDC_BUTTON_SEND, OnSendMessage)
    ON_BN_CLICKED(IDC_BUTTON_ADDR, OnButtonAddrSet)
    ON_BN_CLICKED(IDC_CHECK_TOALL, OnCheckBroadcast)
    ON_WM_TIMER()
    //}}AFX_MSG_MAP
    ON_REGISTERED_MESSAGE( nRegSendMsg, OnRegSendMsg )
    ON_REGISTERED_MESSAGE(nRegAckMsg, OnRegAckMsg)
END_MESSAGE_MAP()
```

- 레지스터에 message를 queue에 보냈다는 신호알림을 저장을 하고 message를 받았다고 알리는 신호 알림을 등록을 하는 단계이다.

```
void CIPCAppDlg::OnSendMessage()
{
    // TODO: Add your control notification handler code here
    UpdateData( TRUE ) ;

    if ( !m_stMessage.IsEmpty() )
    {
        SetTimer( 1, 2000, NULL ) ;
        m_nAckReady = 0 ;

        SendData( ) ;
        m_stMessage = "" ;

        (CEdit*) GetDlgItem( IDC_EDIT_MSG )->SetFocus( ) ;

        ::SendMessage(HWND_BROADCAST, nRegSendMsg, 0, 0);

    }

    UpdateData( FALSE ) ;
}
```

- 프로세스의 인터페이스에 채팅이 입력이 되어지면 등록이 되어진 레지스터를 이용을 하여서 공유 파일에 데이터가 입력이 되었다고 알려주는 함수이다.

```
void CIPCAppDlg::SendData()
{
    CString MsgHeader ;
    if ( m_unDstAddr == (unsigned int)0xff )
        MsgHeader.Format( "[%d:BROADCAST] ", m_unSrcAddr ) ;
    else
        MsgHeader.Format( "[%d:%d] ", m_unSrcAddr, m_unDstAddr ) ;

    m_ListChat.AddString( MsgHeader + m_stMessage ) ;

    int nlength = m_stMessage.GetLength();
    unsigned char* ppayload = new unsigned char[nlength + 1];
    memcpy(ppayload, (unsigned char*)(LPCTSTR)m_stMessage, nlength);
    ppayload[nlength] = '₩0';

    m_ChatApp->Send(ppayload, m_stMessage.GetLength());
}
```

- 프로세스의 인터페이스에 채팅이 입력이 되어지면 broadcast인지 확인을 하고 그 결과에 따라 format을 만들어 주고 데이터를 payload에 저장을 한다. 그 후 ChatAppLayer에 데이터와 메시지의 길이를 전송을 해준다.

```
BOOL CIPCAppDlg::Receive(unsigned char *ppayload)
{
    if ( m_nAckReady == -1 )
    {

        |
    }

    m_ListChat.AddString( (char*) ppayload ) ;

    return TRUE ;
}
```

- IPCAppDlg Layer에 데이터가 도착하면 m_ListChat 변수에 도착한 데이터를 저장을 한다.

```
LRESULT CIPCAppDlg::OnRegSendMsg(WPARAM wParam, LPARAM lParam)
{
    if (m_nAckReady)
    {
        if (m_LayerMgr.GetLayer("File")->Receive())
        {
            ::SendMessage(HWND_BROADCAST, nRegAckMsg, 0, 0);
        }
    }

    return 0 ;
}
```

- 데이터를 받으면 FileLayer의 Receive())함수를 호출을 하고 메시지를 받았다는 nRegAckMsg 신호를 보내준다.

(3) ChatAppLayer.cpp

```
BOOL CChatAppLayer::Send(unsigned char *ppayload, int nlength)
{
    m_sHeader.app_length = (unsigned short) nlength ;

    BOOL bSuccess = FALSE ;
    memcpy(m_sHeader.app_data, ppayload, nlength>APP_DATA_SIZE ? APP_DATA_SIZE : nlength);
    bSuccess = mp_UnderLayer->Send((unsigned char*)&m_sHeader, nlength+APP_HEADER_SIZE);

    return bSuccess ;
}
```

- Send함수는 Ethernet Layer로 데이터를 보내주는 함수로 IPCAppDlg Layer로부터 온 데이터를에 자신에 메시지 헤더를 붙여주고 그 다음 하위 계층인 EthernetLayer로 보내준다.

```
BOOL CChatAppLayer::Receive( unsigned char* ppayload )
{
    PCHAT_APP_HEADER app_hdr = (PCHAT_APP_HEADER) ppayload ;

    if ( app_hdr->app_dstaddr == m_sHeader.app_srcaddr ||
       ( app_hdr->app_srcaddr != m_sHeader.app_srcaddr &&
         app_hdr->app_dstaddr == (unsigned int) 0xff ) )
    {
        unsigned char GetBuff[APP_DATA_SIZE];
        memset(GetBuff, '₩0', APP_DATA_SIZE);
        memcpy(GetBuff, app_hdr->app_data, app_hdr->app_length>APP_DATA_SIZE ? APP_DATA_SIZE : app_hdr->app_length);
        CString Msg;

        if (app_hdr->app_dstaddr == (unsigned int)0xff)
        {
            Msg.Format("[%d:BROADCAST] %s", app_hdr->app_srcaddr, (char*)GetBuff);
        }
        else
        {
            Msg.Format("[%d:%d] %s", app_hdr->app_srcaddr, app_hdr->app_dstaddr, (char*)GetBuff);
        }
        mp_aUpperLayer[0]->Receive((unsigned char*)Msg.GetBuffer(0));

        return TRUE ;
    }
    else
        return FALSE ;
}
```

- Receive() 함수는 EthernetLayer에서 받아온 데이터의 헤더를 검사를 하여 현재 자신의 프로세스의 주소와 비교를 한다. 자신에게 온 데이터가 맞으면 자신의 헤더를 제거한 데이터를 상위 계층인 IPCAppDlg Layer로 전송을 한다.

(4) EthernetLayer.cpp

```
unsigned char* CEthernetLayer::GetDestinAddress()
{
    return m_sHeader.enet_dstaddr;
}
```

- EthernetLayer의 hearder에 들어있는 도착지에 대한 주소 값을 반환을 해주는 함수이다.

```
void CEthernetLayer::SetSourceAddress(unsigned char *pAddress)
{
    memcpy(m_sHeader.enet_srcaddr, pAddress, 6);
}
```

- EthernetLayer의 header의 있는 출발 주소를 저장을 하고있는 장소에 매개변수로 들어온 pAddress의 값을 저장을 한다.

```
BOOL CEthernetLayer::Send(unsigned char *ppayload, int nlength)
{

    memcpy( m_sHeader.enet_data, ppayload, nlength ) ;

    BOOL bSuccess = FALSE ;
    bSuccess = mp_UnderLayer->Send((unsigned char*)&m_sHeader, nlength+ETHER_HEADER_SIZE);

    return bSuccess ;
}
```

- ChatAppLayer에서 전달이 되어져 온 ppayload를 m_sHeader의 data부분의 저장을 하고 하위 계층인 FIleLayer에 m_sHeader를 전송을 해준다.

```
BOOL CEthernetLayer::Receive( unsigned char* ppayload )
{
    PETHERNET_HEADER pFrame = (PETHERNET_HEADER) ppayload ;

    BOOL bSuccess = FALSE ;
    bSuccess = mp_aUpperLayer[0]->Receive((unsigned char*)pFrame->enet_data);

    return bSuccess ;
}
```

- 상위 계층인 ChatAppLayer에서 온 데이터를 EtherLayer의 header를 뗀 부분을 하위 계층인 FileLayer에서 전송을 한다.

(5) FileLayer.cpp

```
CFileLayer::~CFileLayer( )
{
    TRY
    {
        CFile::Remove("IpcBuff.txt");
    }
    CATCH( CFileException, e )
    {
        #ifdef _DEBUG
            afxDump << "File cannot be removed₩n";
        #endif
    }
    END_CATCH
}
```

- 공유파일을 삭제를 하는 역할을 하는 함수이다.

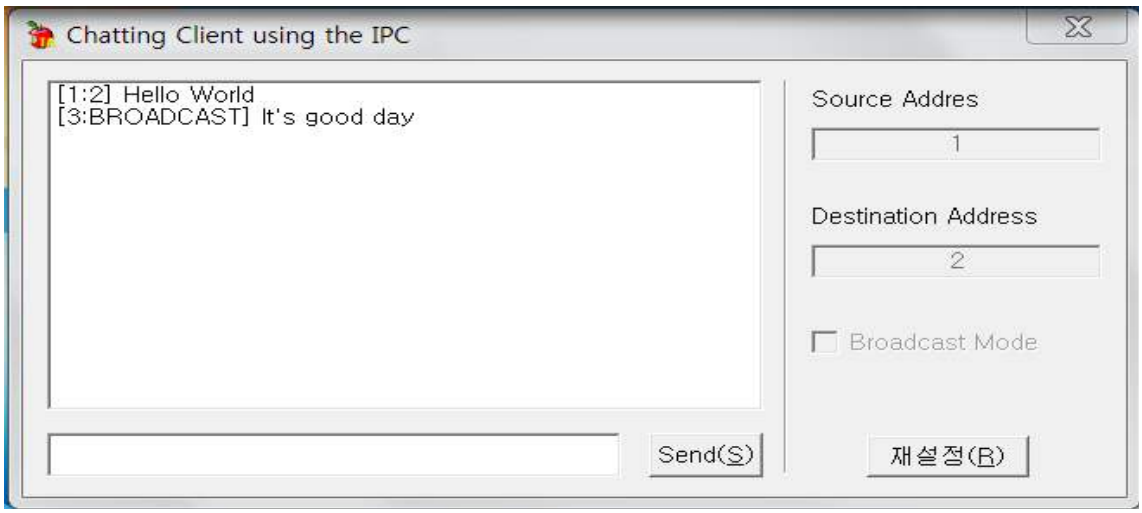
```
BOOL CFileLayer::Send(unsigned char *ppayload, int nlength)
{
    TRY
    {
        CFile m_FileDes( "IpcBuff.txt",
                         CFile::modeCreate | CFile::modeWrite ) ;
    m_FileDes.Write(ppayload, nlength);
    m_FileDes.Close( );
    }
    CATCH( CFileException, e )
    {
#ifdef _DEBUG
          afxDump << "File could not be opened " << e->m_cause << "₩n";
#endif
        return FALSE ;
    }
    END_CATCH

    return TRUE ;
}
```

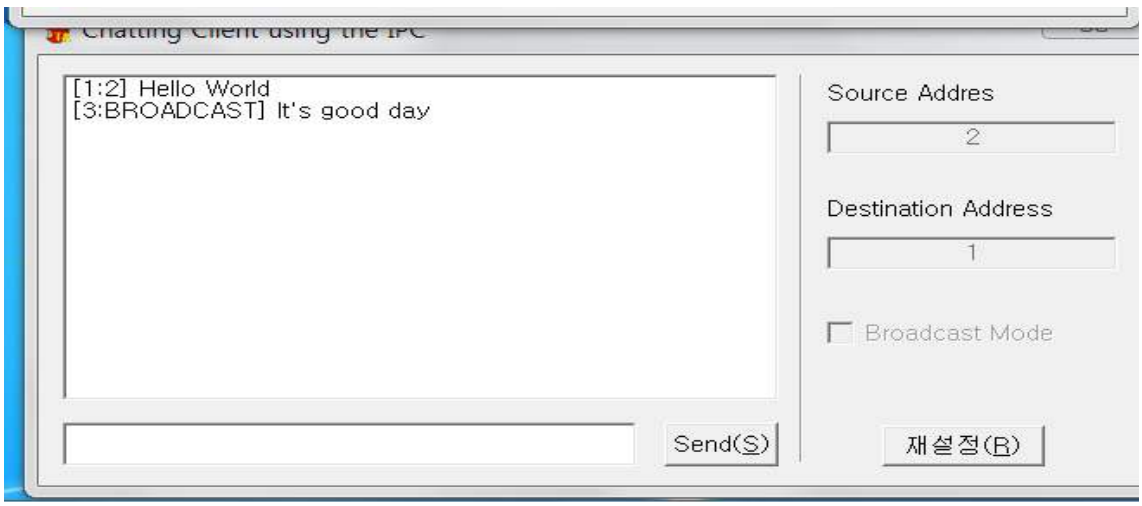
- 이 함수에서는 EthernetLayer에서 전송이 되어져 온 데이터를 공유파일에 쓰는 동작을 하는 함수이다. 공유파일에 데이터를 쓰면서 오류가 일어나면 에러문을 출력을 해준다.

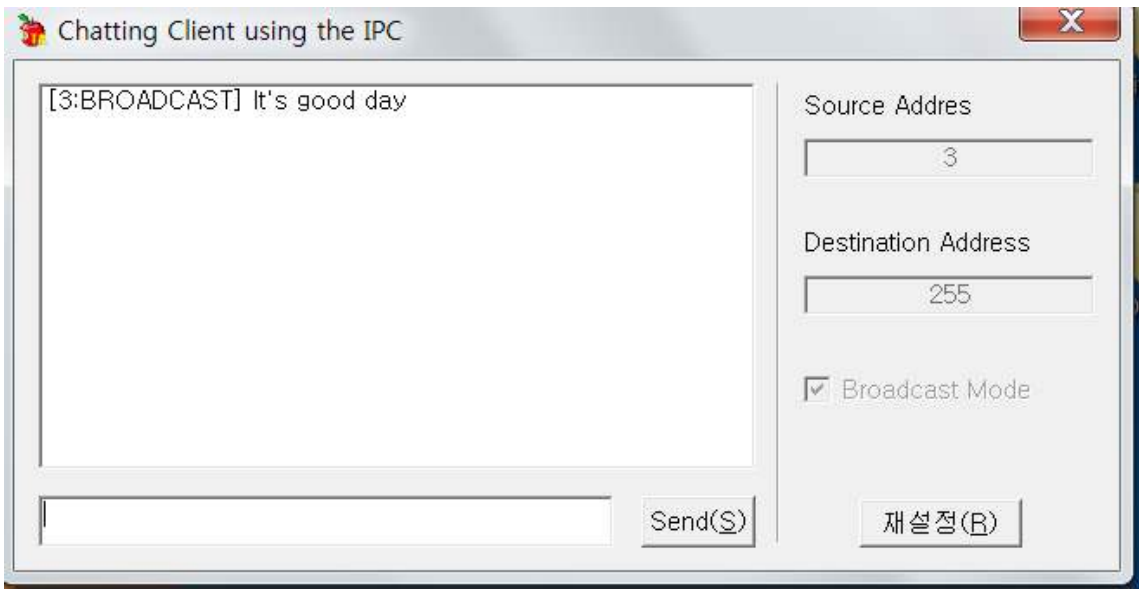
```
BOOL CFileLayer::Receive( )
{
    TRY
    {
        CFile m_FileDes( "IpcBuff.txt", CFile::modeRead ) ;

    int nlength = ETHER_HEADER_SIZE + ETHER_MAX_DATA_SIZE;
    unsigned char* ppayload = new unsigned char[nlength + 1];

    m_FileDes.Read(ppayload, nlength);
    ppayload[nlength] = '₩0';

    if (!mp_aUpperLayer[0]->Receive(ppayload))
    {
        m_FileDes.Close();
        return FALSE;
    }

        m_FileDes.Close( ) ;
    }
    CATCH( CFileException, e )
    {
#ifdef _DEBUG
          afxDump << "File could not be opened " << e->m_cause << "₩n";
#endif
          return FALSE ;
    }
    END_CATCH

    return TRUE ;
}
```

- Receive() 함수에서는 공유파일에 기록이 되어있는 데이터를 읽어서 그것을 상위 계층인 EthernetLayer로 전송을 해주는 역할을 당담을 한다.

4. 실행 결과


Chatting Client using the IPC
[1:2] Hello World
[3:BROADCAST] It's good day
Source Addres
1
Destination Address
2
Broadcast Mode
Send(S)
재설정(R)


[1:2] Hello World
[3:BROADCAST] It's good day
Source Addres
2
Destination Address
1
Broadcast Mode
Send(S)
재설정(R)


Chatting Client using the IPC
[3:BROADCAST] It's good day
Source Addres
3
Destination Address
255
Broadcast Mode
Send(S)
재설정(R)